Using multi-agent systems to go beyond temporal patterns verification

Franco Raimondi, Department of Computer Science
Middlesex University, London
Email: f.raimondi@mdx.ac.uk

A key step in formal verification is the translation of *requirements* into *logic formulae*. Various flavours of temporal logic are commonly used in academia and in industry to capture, among others, *liveness* and *safety* requirements. In the past two decades there has been a substantial amount of work in the area of verification of *extensions of temporal logic*. In this column I will provide a high level overview of some work in this area, focussing in particular on the verification of *temporal-epistemic properties*, showing how temporal-epistemic logics can be used to capture requirements that are common in many concrete systems, and describing a model checker for multi-agent systems called MCMAS.

1. INTRODUCTION

Expressing properties such as mutual exclusion and deadlock freeness using temporal logics such as LTL or CTL [Baier and Katoen 2008] is nowadays considered a standard skill for software engineers and computer science graduates, especially when they are working on safety- and mission-critical projects. Several model checking tools, such as Spin [Holzmann 2003] and NuSMV [Cimatti et al. 2002], support the verification of temporal properties and are now mature enough to be incorporated into the development lifecycle of industrial products. A detailed study of property specifications [Dwyer et al. 1999] has built a repository of patterns, encoding desired *behaviours* of systems in the most common temporal logics.

Temporal logics belong to the more general class of *modal* logics [Hughes and Cresswell 1996], which extend propositional logic with *modalities*. In addition to temporal logics, other examples of modal logics include doxastic logic with modalities to reason about beliefs, deontic logic to reason about obligations, and epistemic logic to reason about knowledge. The proposal of using temporal logic to reason about the correctness of programs has been put forward in [Pnueli 1977] and since then temporal logic has gained wide acceptance in software engineering. However, other modalities have not "transitioned" in a similar way outside the more theoretical venues in Computer Science.

Modal logics have been used routinely in artificial intelligence and multi-agent systems. In particular, there has been a substantial amount of work in the past 20 years in the area of verification for multi-agent systems. On the other hand, the software engineering community and the certification authorities seem to have focussed mainly on temporal specifications. The current trend in integrating automation and humans, for instance in creating autonomous cars and a mixed airspace with autonomous and human-controlled aircraft, gives rise to systems whose requirements can be easily captured by multi-modal logics [Gabbay et al. 2003] that extend temporal-only logic.

In this paper I will focus mainly on temporal-epistemic logic with the aim of showing that the tools and the techniques available can capture in a natural and efficient way several patterns that require reasoning about *knowledge and its temporal evolution* in a system composed of multiple components. I will show how *multi-agent systems* (MAS) provide a suitable abstraction for this kind of reasoning and I will provide an overview of MCMAS [Lomuscio et al. 2015], a model checker dedicated to extensions of temporal logic.

The rest of the paper is organised as follows: I discuss Kripke semantics for extensions of temporal logic in Section 2. I will then present a *computationally grounded*

formalism for multi-agent systems in Section 3, describe the tool MCMAS in Section 4 and introduce some examples in Section 5.

2. KRIPKE SEMANTICS FOR TEMPORAL-EPISTEMIC LOGICS

In this section I introduce the syntax and the semantics of temporal-epistemic logic. I start with a quick overview of temporal logics and then I introduce epistemic logic, discussing the notions of knowledge, group knowledge, distributed and common knowledge.

2.1. Temporal logics

The two most common flavours of temporal logic are LTL (Linear Temporal Logic) and CTL (Computation Tree Logic). The syntax of LTL is as follows:

$$\phi ::= \quad p \quad | \quad \neg\phi \quad | \quad \phi \vee \phi \quad | \quad X\phi \quad | \quad G\phi \quad | \quad F\phi \quad | \quad \phi U \phi$$

where the other propositional connectives for conjunction and implication can be derived as usual. The temporal operators X, G, F express, respectively, that something is true in the neXt state, Globally, and at some point in the Future. The binary operator U requires that the second formula ψ must become true at some point in the future, and until that point ϕ has to remain true. The syntax of CTL is similar, but it prefixes each temporal operator with a *path quantifier*:

$$\phi ::= \quad p \quad | \quad \neg\phi \quad | \quad \phi \vee \phi \quad | \quad EX\phi \quad | \quad EG\phi \quad | \quad EF\phi \quad | \quad E[\phi U \psi]$$

(the dual quantifier A for "All paths" can be obtained in a standard way).

Given a set of atomic propositions AP, temporal logic formulae are interpreted in *Kripke structures* $M = (S, t, L, I)$, where S is a set of states, $t \subseteq S \times S$ is a temporal transition relation, $L : S \to 2^{AP}$ is a labelling function for states, and $I \subseteq S$ is a set of initial states. In the reminder, I will use the term Kripke structure and *model* interchangeably. A path π is a sequence of states $\pi = (s_0, s_1, \dots)$ such that $(s_i, s_{i+1}) \in t$ for all $i \geq 0$. A state s_i in a path π is denoted by $\pi(i)$. CTL formulae are evaluated in a state of a model, while LTL formulae are evaluated over *paths*. In particular, the semantics of temporal CTL operators is defined by (Boolean connectives being defined in an obvious way in terms of the labelling function L)[1]:

$$\begin{aligned}
M, s \models EX\phi &\text{ iff there exists a path } \pi \text{ s.t. } s = \pi(0) \text{ and } M, \pi(1) \models \phi \\
M, s \models EG\phi &\text{ iff there exists a path } \pi \text{ s.t. } s = \pi(0) \text{ and } M, \pi(i) \models \phi \text{ for all } i \geq 0 \\
M, s \models E[\phi U \psi] &\text{ iff there exists a path } \pi \text{ s.t. } s = \pi(0) \text{ and there exists a } j \text{ s.t.} \\
&\qquad M, \pi(j) \models \psi \text{ and for all } 0 \leq i \leq j, M, \pi(i) \models \phi
\end{aligned}$$

The semantics of LTL operators is defined in terms of paths, as follows:

$$\begin{aligned}
M, s \models X\phi &\text{ iff } M, \pi[i] \models \phi \\
M, s \models G\phi &\text{ iff } M, \pi[i] \models \phi \text{ for all } i \geq 0 \\
M, s \models [\phi U \psi] &\text{ iff there exists a } j \text{ s.t. } M, \pi[j] \models \psi \text{ and for all } 0 \leq i \leq j, M, \pi[i] \models \phi
\end{aligned}$$

(where $\pi[i]$ is the suffix path of π starting at state i).

2.2. Epistemic and temporal-epistemic logics

The syntax of single-agent epistemic logic is given by the following grammar:

$$\phi ::= \quad p \quad | \quad \neg\phi \quad | \quad \phi \vee \phi \quad | \quad K\phi$$

[1]The semantics of EF can be defined in term of EU, as $EF\phi \equiv E[\top U \phi]$.

in which propositional logic operators are extended with the operator K. The formula $K\phi$ is read as "knows that ϕ" and is normally used to express the epistemic state of an *agent* (more on this later).

Epistemic formulae are interpreted in Kripke structures $M = (S, \sim, L)$ in which, as in the previous section, S is a set of states, L is a labelling function and $\sim \subseteq S \times S$ is an *accessibility relation* that is serial, transitive and symmetric. The accessibility relation $\sim$ is thus an equivalence relation encoding the set of states that are *indistinguishable* for an agent. The semantics of the K operator is given in terms of $\sim$, as follows:

$$M, s \models K\phi \text{ iff } M, s' \models \phi \text{ for all } s' \text{ s.t. } (s, s') \in \sim$$

Intuitively, an agent consider $K\phi$ true in a state s if and only if ϕ is true in all the states that the agent cannot distinguish form s. This is a very strong notion of "knowledge": not only it implies that an agent knows all tautologies, it also implies positive and negative introspection, in the sense that, if the agents knows something, then it knows that it knows it, and if an agent *does not know* something, then the agent *knows that it does not know it*. Nevertheless, this characterisation of knowledge is appropriate for a range of scenarios [Fagin et al. 2003], such as when reasoning about security properties and other epistemic states of agents.

Temporal and epistemic logics can be *fused* (in a technical sense, see [Gabbay et al. 2003]), resulting in a multi-dimensional modal logic in which the temporal and epistemic components are evaluated separately. In fact, one could consider multiple agents, each of them with their own epistemic accessibility relation. Consider n agents: a temporal-epistemic Kripke structure is defined as

$$M = (S, t, \{\sim_i\}_{1 \leq n}, L)$$

This Kripke structure has $n + 1$ relations: a temporal one and n epistemic accessibility relations, one for each agent. This model allows the interpretation of formulae defined by the following syntax, in addition to propositional and temporal operators:

$$\phi ::= K_i\phi \quad | \quad E_\Gamma\phi \quad | \quad D_\Gamma\phi \quad | \quad C_\Gamma\phi$$

where i is an index ranging from 1 to n, $\Gamma \subseteq \{1, \ldots, n\}$ is a group of agents, $E_\Gamma\phi$ denotes that *everybody* in a group knows ϕ, $D_\Gamma\phi$ denotes that ϕ is distributed knowledge in the group, and $C_\Gamma\phi$ denotes that ϕ is common knowledge in the group. To clarify the difference between these three group operators, consider the following example: Alice is the teenage daughter of Bob. Alice smokes cigarettes but she has always kept this secret from her father, so she thinks that he does not know that she smokes. However, one day Bob could see Alice smoking in a corner, without being noticed by her. Additionally, Alice keeps cigarettes in a drawer in her bedroom, and Bob does not know about this. In this situation, everybody knows that Alice smokes. Also, the location of the cigarettes is distributed knowledge between Alice and Bob (this is the knowledge that could be achieved if epistemic states where shared). However, the fact that Alice smokes is not *common knowledge* because Alice does not know that her father knows that she smokes. Suppose now that one day Bob enters Alice's room while she is smoking: at this point the fact that Alice smokes is *common knowledge* between Alice and Bob. I refer to [Fagin et al. 2003] for the formal semantics of epistemic group operators in terms of the epistemic accessibility relations of each agent.

3. MULTI-AGENT SYSTEMS

One of the reasons for temporal patterns to be more widely used than epistemic or other modalities is probably the clear link between the temporal evolution of a program

and the temporal relation t defined in the previous section. A possible execution of a program path *is* a possible path in a model, and the set of all possible executions *is the model*. To make verification of temporal-epistemic properties viable it is necessary to make clear the link between "real" executions and Kripke structures. This is exactly the notion of *computationally grounded semantics* introduced in [Wooldridge 2000].

3.1. Interpreted systems

In this section I describe the formalism of *interpreted systems* [Fagin et al. 2003], showing how they link both to executions of "real" systems and to Kripke structures. For a given set of n agents $Agt = \{Ag_1, Ag_2, \dots Ag_n\}$, an interpreted system is a tuple:

$$IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Agt}, I, h)$$

where L_i is a finite set of local states for agent Ag_i, Act_i is a finite set of actions that agent Ag_i can perform, $P_i : L_i \rightarrow 2^{Act_i}$ is a function, called a protocol, that specifies which actions are enabled in each local state for agent Ag_i, and $t_i : (L_i \times Act_1 \times \dots \times Act_n) \rightarrow L_i$ is a local temporal relation that describes the evolution of the local states of Ag_i in terms of its current local states and of the actions of *all* the other agents. This implies that agents can only "see" the actions of the other agents, but not their local states. The characterisation of an interpreted system is completed by a set of initial states $I \subseteq L_1 \times \dots \times L_n$ and by an evaluation function $h : (L_1 \times \dots \times L_n) \rightarrow 2^{AP}$, where AP is a set of atomic propositions.

Each interpreted system gives rise to an *induced* multi-dimensional Kripke structure $M_{IS} = (S, t, \{\sim_i\}_{i \in Agt}, L)$, as follows:

— The set S is the set of of *reachable global states*: this is the set of states $S \subseteq (L_1 \times \dots \times L_n)$ that are reachable from I according to the local temporal relations.
— The temporal relation t is defined by the composition of the temporal relations t_i that, in turn, take into account the protocols of each agent.
— The epistemic accessibility relations are defined by equivalence of local states. Formally, two global states $g = (l_1, \dots, l_n)$ and $g' = (l'_1, \dots l'_n)$ are epistemically equivalent for agent i iff $l_i = l'_i$. This is a key point, as it allows to define epistemic accessibility relations as the byproduct of the temporal evolution and the structure of the overall system, thereby providing a *computationally grounded* semantics.
— The evaluation function L is identical to the evaluation function h.

As a result, it is possible to use all the known results for standard Kripke structures in interpreted systems. In particular, traditional model checking algorithms can be employed for the verification of temporal-epistemic formulae in interpreted systems, as discussed in the next section.

3.2. Model checking multi-agent systems

Several approaches have been developed to verify multi-agent systems using model checking. In general, and at a very high level, most of the model checking techniques [Baier and Katoen 2008] for temporal logics transfer to model checking multi-agent systems. Examples of model checkers for multi-agent systems include:

— VerICS [Kacprzak et al. 2008; Meski et al. 2011]: this is a model checker that support the verification of CTL and LTL extended with knowledge. It implements model checking techniques based on Ordered Binary Decision Diagrams (OBDDs [Bryant 1986]) and reduction to SAT for bounded model checking.
— MCK [Gammie and van der Meyden 2004; MCK 2016]: this is a model checker that supports the verification of epistemic and CTL* properties, including support for *per-*

```
Agent SampleAgent
  Vars:
    v1 : { a,b,c }; -- This is a comment
    v2 : boolean;
  end Vars
  Actions = { action1, action2, action3 };
  Protocol:
    v1=a and v2=false : {action1};
    v1=b : {action2,action3};
    Other : {action3};
  end Protocol
  Evolution:
    (v2=true) if (v2=false) and
        (AnotherAgent.Action=someAction) ;
  end Evolution
end Agent
```

Fig. 1. A minimal ISPL example: an Agent.

fect recall semantics [van der Meyden and Shilov 1999] for some specific patterns of formulae.

— Other approaches have reduced the model checking problem for multi-agent system to a temporal-only model checking problem using NuSVM [Su et al. 2007; Lomuscio et al. 2007].

In this column I will focus mainly on the model checker MCMAS [Lomuscio et al. 2015]. This model checker supports the verification of CTL properties extended with epistemic operators (both for individual and for group of agents), in addition to ATL operators [Alur et al. 2002], implementing OBDD-based verification. MCMAS implements the standard model checking algorithm for CTL, which is based on the fix-point characterisation of temporal operators [Baier and Katoen 2008]. This algorithm is extended to epistemic operators for individual agents by taking into account that the K operator behaves like the AX operator, but over the epistemic transition. The operator for common knowledge, instead, is computed as a fix-point of the operator for "everybody knows" E_Γ , building on the observation that $C_\Gamma\phi \equiv E_\Gamma(\phi \wedge C_\Gamma\phi)$.

4. THE MODEL CHECKER MCMAS

MCMAS [Lomuscio et al. 2015] is a Model Checker for Multi-Agent Systems, released for the first time in 2004. It allows the verification of CTL formulae extended with epistemic modalities (both individual and groups), the verification of ATL properties [Alur et al. 2002] and it supports fairness conditions and counter-example/witness generation. In its default configuration MCMAS implements the standard labelling algorithm for CTL extended to epistemic and strategic operators. The verification engine makes use of OBDDs [Bryant 1986] to represent states and transitions symbolically.

MCMAS semantics is based on interpreted systems and its input language is called ISPL (Interpreted Systems Programming Language). Figure 1 provides an example to encode an agent called `SimpleAgent`. The local states of an agent are defined by the variables it controls: in this example, two variables are declared, an enumeration and a boolean variable, for a total of 6 possible local states. The data types supported by MCMAS are boolean, enumerations, and bounded numeric ranges. Actions are defined as a simple list of labels, while the protocol section allows the introduction of non-determinism in the model (see the protocol line assigning two possible actions to the case `v1 = b`). The evolution function is encoded using a NuSMV-like syntax: the evolution line in Figure 1 is read as "The *next* value of `v2` is true if the current value is false and the action of `AnotherAgent` is `someAction`."

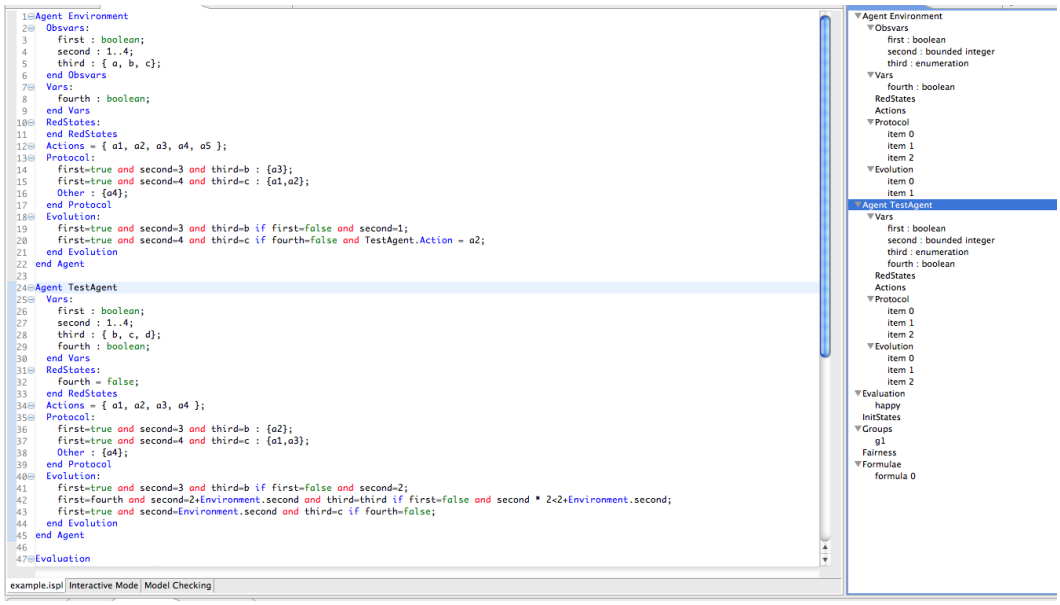

Fig. 2. MCMAS GUI: editor window (taken from [Lomuscio et al. 2015])

MCMAS is available for download from http://vas.doc.ic.ac.uk/software/mcmas/. The tool is released as open source and it includes a graphical interface based on an Eclipse plug-in. A screenshot of the editor window is shown in Figure 2. Formulae to be verified are listed at the end of the ISPL file and the tool can generate a witness/counter-example for temporal *and* epistemic/strategic formulae. These are displayed as a tree and can be exported as an image. The tool is built using standard C/C++, Flex and Bison and it has been compiled on Windows, Mac, and Linux. I refer to [Lomuscio et al. 2015] for a more detailed description and for performance considerations.

5. EXAMPLES

As mentioned at the beginning of this column, the key feature of temporal-epistemic logic is that it allows the encoding of several requirements that may be difficult to express using temporal-only formulae. Examples of domains in which temporal-epistemic logic have been employed are the verification of authentication protocols [Boureanu et al. 2009], the verification of BPEL web-service composition [A. Lomuscio 2012], the verification of a confidential conference management system [Kanav et al. 2014], etc.

In this section I provide two simple examples to show how temporal-epistemic logic can express the key properties of two systems in a compact and intuitive form: the protocol of the dining cryptographers [Chaum 1988] and a characterisation of *situational awareness* using a temporal-epistemic pattern [Chen et al. 2015].

5.1. The protocol of the Dining Cryptographers

The original wording of this example is as follows:

"Three cryptographers are sitting down to dinner at their favourite three-star restaurant. Their waiter informs them that arrangements have been made with the maitre d'hotel for the bill to be paid anonymously. One of the cryptographers might be paying for the dinner, or it might have been NSA (U.S. National Security Agency). The three cryptographers respect each other's right to make an anonymous payment, but they won-

der if NSA is paying. They resolve their uncertainty fairly by carrying out the following protocol:

Each cryptographer flips an unbiased coin behind his menu, between him and the cryptographer on his right, so that only the two of them can see the outcome. Each cryptographer then states aloud whether the two coins he can see – the one he flipped and the one his left-hand neighbour flipped – fell on the same side or on different sides. If one of the cryptographers is the payer, he states the opposite of what he sees. An odd number of differences uttered at the table indicates that a cryptographer is paying; an even number indicates that NSA is paying (assuming that the dinner was paid for only once). Yet if a cryptographer is paying, neither of the other two learns anything from the utterances about which cryptographer it is."[Chaum 1988]

This protocol is the basic building block for the definition of more complex infrastructures enabling anonymous communication over public networks; moreover, notice that the protocol works for any number of cryptographers greater or equal to three.

The key property of this example can be formalised by the following formula

$$AG\Big(\Big(\bigwedge_{i=1}^{n} c_i\_announced \wedge \neg c_1\_paid\Big) \rightarrow \Big(K_{c_1}(\bigwedge_{i=1}^{n} \neg c_i\_paid) \vee \Big(K_{c_1}(\bigvee_{i=2}^{n} c_i\_paid) \wedge \bigwedge_{i=2}^{n} \neg K_{c_1}(c_i\_paid)\Big)\Big)\Big)$$

where AG is the CTL temporal operator expressing that the formula is true in all states, n is the number of cryptographers, $c_i\_announced$ represents the announcement made by cryptographer i, and c_i_paid encodes the fact that cryptographer i paid the bill. Anonymity is captured by the fact that, if a cryptographer did not pay and there is an odd number of "different" utterances, then the cryptographer *knows* that someone paid for the dinner (expressed as a disjunction), but the cryptographer does not know who actually paid.

5.2. Situational awareness as a temporal-epistemic formula

As stated in [Chen et al. 2015], *"[i]nformally, situational awareness is the ability of an agent (typically human) to determine the correct internal state of some component (or some other agent) based on his/her current beliefs. Situational awareness is a key factor for decision makers in safety-critical situations, such as airplane pilots, medical doctors, firemen, etc"*.

The work in [Chen et al. 2015] introduces an extension of epistemic logic that allows to *count* the number of states of an epistemic equivalence class in which a given formula is true. In turn, this allows for the definition of a doxastic operator $B^i_{<\delta}\phi$, which is true if the ratio of states in an equivalence class in which ϕ is true is $< \delta$. The work in [Chen et al. 2015] models an avionic scenario and it defines the *lack* of situational awareness of a situation ϕ if there exists a state in which ϕ is true, but an agent believes it to be true with certainty less than 5% (this limit is arbitrary and can be modified). Specifically, consider a stall situation characterised by the proposition actualStall, and consider that a Pilot can be represented by an agent. Then, the fact that it is possible for a pilot to have lack of situational awareness of a stall can be characterised by the following formula:

$$EF\left(\text{actualStall} \wedge B^{\text{Pilot}}_{<.05}\left(\text{actualStall}\right)\right)$$

A tool like MCMAS can be used to verify whether this formula is true in a model of the cockpit of an airplane, thus enabling the detection of potentially dangerous flaws very early in the design stages of a project.

6. CONCLUSION

In this column I have discussed how the extension of temporal logic with epistemic operators enables the formalisation of requirements from a range of domains. After introducing some basic concepts, I have discussed the notion of computationally grounded semantics for multi-agent systems and I have presented MCMAS, a tool that supports the verification of extensions of the temporal logic CTL in multi-agent systems.

My opinion is that an understanding of extensions of temporal logic should be part of the skills of everyone involved in verification of critical systems, as these extensions enable the formalisation of a range of requirements that may be difficult to express otherwise. I have only sketched some of the research in this area and I refer the interested reader to the proceedings of conferences such as AAMAS and IJCAI for more in-depth results.

Acknowledgements

I gratefully acknowledge Giuseppe Primiero for his comments and feedback.

REFERENCES

M. Solanki A. Lomuscio, H. Qu. 2012. Towards verifying contract regulated service composition. *Autonomous Agents and Multi-Agent Systems* 24, 3 (2012), 345–373.

R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *J. ACM* 49, 5 (2002), 672–713.

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

I. Boureanu, M. Cohen, and A. Lomuscio. 2009. A compilation method for the verification of temporal-epistemic properties of cryptographic protocols. *Journal of Applied Non-Classical Logics* 19, 4 (2009), 463–487.

R. Bryant. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transaction on Computers* 35, 8 (1986), 677–691.

D. Chaum. 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology* 1 (1988), 65–75.

Taolue Chen, Giuseppe Primiero, Franco Raimondi, and Neha Rungta. 2015. A Computationally Grounded, Weighted Doxastic Logic. *Studia Logica* (2015), 1–25.

Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. 2002. NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV '02)*. Springer-Verlag, London, UK, UK, 359–364. http://dl.acm.org/citation.cfm?id=647771.734431

Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1999. Patterns in Property Specifications for Finite-state Verification. In *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*. ACM, New York, NY, USA, 411–420. DOI: http://dx.doi.org/10.1145/302405.302672

Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. 2003. *Reasoning About Knowledge*. The MIT Press.

Dov M Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. 2003. *Many-dimensional modal logics: theory and applications*. Vol. 148. North Holland.

P. Gammie and R. van der Meyden. 2004. MCK: Model Checking the Logic of Knowledge. In *Proceedings of CAV 2004 (Lecture Notes in Computer Science)*, Vol. 3114. Springer, 479–483.

Gerard Holzmann. 2003. *Spin Model Checker, the: Primer and Reference Manual* (first ed.). Addison-Wesley Professional.

G.E. Hughes and M. J. Cresswell. 1996. *A New Introduction to Modal Logic*. Routhledge, London.

M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Wozna, and A. Zbrzezny. 2008. VerICS 2007 - a Model Checker for Knowledge and Real-Time. *Fundamenta Informaticae* 85, 1-4 (2008), 313–328.

Sudeep Kanav, Peter Lammich, and Andrei Popescu. 2014. *Computer Aided Verification: 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*. Springer International Publishing, Cham, Chapter A Conference Management System with Verified Document Confidentiality, 167–183. DOI:http://dx.doi.org/10.1007/978-3-319-08867-9_11

A. Lomuscio, C. Pecheur, and F. Raimondi. 2007. Automatic Verification of Knowledge and Time with NuSMV. In *Proceedings of IJCAI07*. 1384–1389.

Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. 2015. MCMAS: an open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer* (2015), 1–22. DOI:http://dx.doi.org/10.1007/s10009-015-0378-x

MCK 2016. MCK. (2016). http://cgi.cse.unsw.edu.au/ mck/pmck/.

Artur Meski, Wojciech Penczek, and Agata Pólrola. 2011. BDD-based Bounded Model Checking for Temporal Properties of 1-Safe Petri Nets. *Fundamenta Informaticae* 109, 3 (2011), 305–321.

A. Pnueli. 1977. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*. 46–57. DOI:http://dx.doi.org/10.1109/SFCS.1977.32

K. Su, A. Sattar, and X. Luo. 2007. Model Checking Temporal Logics of Knowledge Via OBDDs. *Computer Journal* 50, 4 (2007), 403–420.

R. van der Meyden and N. Shilov. 1999. Model Checking Knowledge and Time in Systems with Perfect Recall. In *Conf. on Foundations of Software Technology and Theoretical Computer Science (Lecture Notes in Computer Science)*, Vol. 1738. Springer, 432–445.

M. Wooldridge. 2000. Computationally Grounded Theories of Agency. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, E. Durfee (Ed.). IEEE Press.